VŠB - Technická univerzita Ostrava
Fakutla elektrotechniky a informatiky
Katedra informatiky

Generování grafiky pro živé televizní vysílání

Generation of Info graphics for On-line Video Streaming

2014 **Marek Urban**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Marek Urban**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Generování grafiky pro živé televizní vysílání
Generation of Info graphics for On-line Video Streaming

Zásady pro vypracování:

Cílem práce je vytvořit systém pro generování grafiky pro on-line sportovní přenosy. Aplikace bude zejména generovat stavy hry, herní statistiky, informační grafiku apod. Výstup aplikace se bude připojovat do video režie, kde se bude kombinovat s dalšími signály. Aplikace bude umět:
1. Měnit data v grafice v reálném čase.
2. Definovat několik typů přednastavené grafiky, mezi kterými je možné rychle přepínat.
3. Nastavit pozici jednotlivých prvků grafiky, dle okolností.
4. Ukládat a načítat jednotlivé profily.
5. Ukládat informace o hře, hráčích a jejich historii pro potřeby dalšího využití
6. Načítat data z externích zařízení jako jsou výsledkové tabule apod, ale v případě absence, umožnit ruční zadání/změnu hodnot.

V práci musí být analýza případných existujících řešení, analýzu vlastní aplikace a návrh řešení a případnou strukturu vytvořené databáze.

Práce bude realizována v jazyce C# s pomocí podpůrných technologií.

Seznam doporučené odborné literatury:

[1] Jay Glynn a kol.: C# Programujeme profesionálně, COMPUTER PRESS, ISBN: 9788025100851

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Platoš, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2014

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA * Fakulta elektrotechniky a informatiky * OSTRAVA *

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: *17.7.2014*

……..……
podpis studenta

Prohlášení zástupce spolupracující právnické nebo fyzické osoby

„Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.“

Dne: *17.7.2014*

……..……
podpis zástupce

Poděkování

Rád bych poděkoval *doc. Ing. Janu Platošovi Ph.D.* za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Cílem práce bylo vytvořit cenově dostupné řešení problému interaktivní grafiky v živém televizním přenosu. Idea zněla, vytvořit aplikaci, která bude na standartní (ale i nestandartní) grafický výstup poskytovat uživatelem zvolenou grafiku na klíčovatelném pozadí. Současně musí být aplikace schopna pracovat s dynamickými daty. Těmito daty se především myslí sportovní časomíry, externí statistiky, bodové stavy a podobné. Aby bylo možné aplikaci efektivně použít pro živé vysílání, musí aplikace umožňovat okamžitou změnu dat, bez nutnosti upravovat samotnou grafiku.

Klíčová slova

Televizní vysílání, grafika, datové zdroje, klíčování, generování obrazu, overlay, alfa kanály, .net Framework, wpf aplikace

Abstract

The main purpose of this thesis was to create a low cost solution for interactive graphics in live television broadcast. The idea was to create an application which is capable of generating video stream containing graphics on keyable background (solid color). At the same time the application must be able to work with dynamic data. Dynamic data are data that are being fetched by external or internal data source into the graphics and change over time period. By external data sources we mean for example score boards, databases or socket servers. On the other hand the internal data source is for example a timer. For the effective usage in live production the application must be able to change data inside the graphics without need to modify graphics alone.

Key words

Television broadcast, graphics, data sources, keying, image generation, overlay system, alpha channel, .net framework, wpf application

Seznam zkratek a pojmů

Zkratka / Pojem	Anglický význam	Český význam
SD	Standard definition	Standartní rozlišení - klasický televizní formát 720x576 px
HD	High definition	Vysoké rozlišení (až 1920x1080px)
FPS	Frames per second	Snímky za vteřinu
Datové zdroje	Data Sources	Prvek programu poskytující specifický typ dat.
Grafický prvek	Graphics	Prvek aplikace reprezentující konkrétní grafickou entitu zobrazovanou ve videu.
Datové pole	Data Label	Entita grafického prvku zobrazuje data z datových zdrojů.

Obsah

1. Úvod

Textové informace, doprovázené grafikou, jsou dnes již nedílnou součástí každého audiovizuálního díla. U postprodukční práce (ne-živé prudukce) je vkládání grafických a textových informací do videa řešeno střihovým softwarem. U živé produkce tedy nastává problém jak grafický obsah do obrazu dostat.

Základem každé živé video produkce je střižna, která promíchává a přepíná video signály z externích vstupů (kamery, video servery, zdroje grafiky a další). Výrobci současných střihových systémů samozřejmě na problém grafikých zdrojů mysleli a určité grafické generátory do svých zařízení integrovali nebo jen umožnili připojení vlastních PC. Bohužel intergrované grafické generátory mají pouze velmi drahé televizní komplety. Příkladem takového kompletu může být například BroadcastPix Granite (používaný například v přenosových vozech ČT Sport), jehož cena se pohybuje od 30 000 amerických dolarů nahoru. Nicméně i u toho televizního systému není možné načítat data z externích zdrojů, jeho možnosti jsou omezené jen na zobrazování jednoduchých titulků do obrazu.

Pro odbavování sportovních událostí je potřeba systém, který zvládá, zobrazování času (herní čas, přesilovky), dynamických externích dat jako jsou statistiky, mezičasy, tresty nebo například data z cedulí. Konkrétně ČT Sport tento problém řeší naklíč vytvořeným systémem, který k “čistému” obrazu ze střižny domíchává grafiku s daty pomocí alfa kanálu v grafice. Tyto systémy se označují jako overlay, většinou se jedná o výkonný obrazový procesor, který s patřičnou aplikací zajišťuje vložení grafických prvků do videa. Nicméně i tyto systémy jsou pro běžné smrtelníky (měnší studia s kapitálem menším než milion Kč) nedostupné, hlavně kvůli astronomickým pořizovacím cenám, (ČT za svůj systém zaplatila 6 milionů Kč).

Alternativou k drahým řešením je použít technologii klíčování. Schopnost klíčovat, tedy nahradit konkrétní barvu v obraze alfa kanálem, mají dneska i již ty nejlevnější střižny, tedy je to stále často volené řešení.

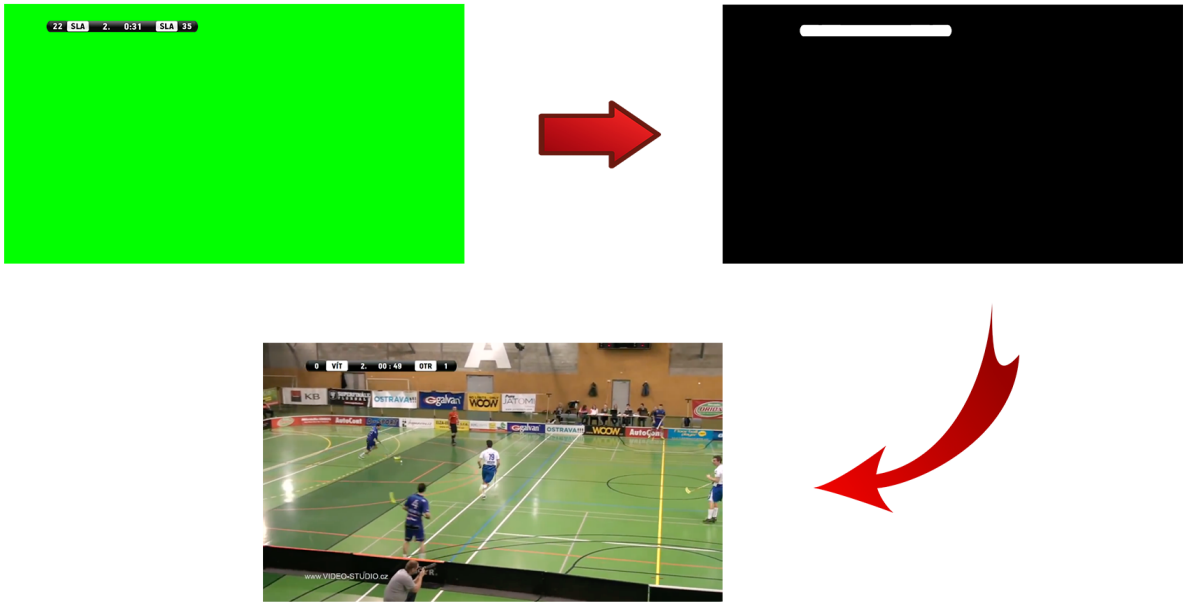
2. Současné technologie

1. Overlay systémy

Overlay jako modernější a pokročilejší technologie se opírá především o silný výkon grafických procesrů (GPU) které jsou schopny manipulovat s matricí videa v realném čase. Při nedostatku výkonu tak může nastat zaseknutí jak grafiky tak i celého video streamu (jako se často děje u levnějších overlay systémů), při selhání systému pak dokonce hrozí naprostý výpadek signálu. Další nevýhodou těsto systému je fakt, že musí být spraženy s řídící aplikací, která jasně vymezuje co celý systém umí a neumí, což zřídka kdy umožnuje uživateli použití dalších rozšíření a vlastních nádstaveb či řešení.

2. Klíčování

O klíčování se stará samotná střižna, případně speciální klíčovací modul, který nijak není spřažen s aplikací generující grafiku. Je tedy možné použití vlastních aplikací, nebo případně hotových obrázku z Photoshopu, či prenzetací z Powerpointu a dalších řešení. Při selhání generátoru grafiky nedochází k výpadku celého signálu. Nikdy tedy nemůže dojít k degradaci kvality obrazu z důvodu vysokého zatížení grafické jednotky, jelikož se o samotnou implementaci grafických prvků do videa stará oddělené zařízení. Klíčování ovšem přináší i jistá omezení. Předem je třeba vybrat barvu která se nesmí v grafice vyskytovat, tuto barvu bude mít pozadí, které bude nahrazeno alfa kanálem a následně samotným obrazem. Dnešní klíčovací moduly jsou natolik kvalitní, že dokáží definovat klíčovanou barvu velmi přesně, stačí tedy s problematickou barvou lehce “uhnout” (např. ztmavit, zesvětlit, či změnit odstín).



Obrázek 2.1: Princip klíčování

3. Současná řešení

V mé celkově 7-mi leté praxi v oboru jsem se setkal s několika řešeními problému živé televizní grafiky. Ať už šlo o "po domácku" udělané řešení využívající aplikace jako Adobe Photoshop, Microsoft PowerPoint nebo dražší hotové systémy jako NewTek Tricaster. V posledních letech jsem měl možnost pracovat i s televizními sytémy České Televize (stroje Broadcast Pix). Zde je tedy porovnání výhod a nevýhod těchto řešení.

Oddělené softwarové řešení

Výrobci levnějších střižen jako podporují připojení klasického počítačového výstupu (HDMI, DVI). Je tedy na samotném zákazníkovi jakou softwarovou výbavou si počítač (generátor signálu) osadí.

Výhody:
- Vysoká optimalizovatelnost - řešení specifických problémů
- Relativně nízká pořizovací cena - není potřeba speciální hardware

Nevýhody:
- Nutnost využití klíčování - po grafickém výstupu nelze poslat video s alfa kanálem, je tedy nutnost mít video s jednobarevným pozadím, které si střižna následně odstraní.
- Problémy s částečnou průhledností - absence alfa kanálu ve videu způsobuje nejasnou definovatelnost průhledných části grafiky, což vede k barevnému zkreslení obrazu (spíše u starších střižen a klíčovacích modulů)
- Špatná kontrola nad grafickým vstupem - střihač nemá možnost si zvolit, kterou grafiku chce zrovna zobrazit případně skrýt (například přehrávání opakovaného záběru vyžaduje stažení klasické herní grafiky a zobrazení například nápisu Replay. Tento úkon musí provést obsluha grafického generátoru a musí ho provést synchronně s videem, což může být (a je) v živé produkci náročné. Dalším příkladem by mohlo být zobrazení titulku s hráčem, titulek se musí zobrazit přesně v době střihu správného záběru a zmizet dřív než daný záběr skončí.

Příklady zařízení:
BlackmagicDesign Atem series - Střižny ATEM od firmy BlackmagicDesign umožnují vkládat grafiku do videa přes Adobe Photshop, střižna je s PC propojená Ethernetem nejde tedy o klasický přenos video signálu, ale čistých souborů (podpora alfa kanálu). Střižny taktéž disponují klasickým vstupem HDMI pro připojení grafické karty počítače.

Mixážní pulty DataVideo - podporují připojení počítače přes grafickou kartu (střižna se hlásí jako externí monitor). Často se používá v kombinaci s MS Powerpoint pro zobrazování sestav hráčů, časového harmonogramu vysílané akce a podobně.

Oba řešení neumožňují vložení například běžící časomíry, umí tedy zobrazovat pouze předem připravená statická data.

Implementované softwarové řešení

Americký výrobce NewTek už 10 let vyrábí mofikované pracovní stanice, které jsou kompletními střižnami a zároveň obsahují možnosti vytváření grafických prvků do videa, přímo ve střižně během živé produkce.

Výhody:
- Kvalitní vložení grafiky do videa pomocí overlay technologie.
- Střihač má pod kontrolou přesně jakou grafiku do videa pouští a s jakými daty. Snížení chyb během produkce
- Široké možnosti využití efektů střižny

Nevýhody:
- Pořizovací cena (nejlevnější model televizního standardu je Tricaster 410 stojí 10 000 USD)
- Nemožnost optimalizace - funkce jsou závislé na výrobci
- Dynamická data řešena jen částečně - systém umožňuje měnit texty v grafice během zobrazení grafiky ve videu, nicméně neumožnuje například vložit do grafiky časomíru (kterou by řídila střižna)
- Načítání z externích zdrojů - je umoženěno pomocí DataLink modulu (ten umožňuje i zobrazení časomíry). Nicméně modul je kompatibilní jen s malým množstvím výrobců sportovních cedulí, kteří zpravidla figurují jen v USA.

Obrázek 2.2: Střižna TriCaster 8000

Dedikovaná zařízení (Match moving)[1]

Nejdražší a nejpokrokovější technologií jsou počítačové grafické systémy, se schopnosti manipulování s obrazem v realném čese. Využívají jak technologii klíčování (pro definování objektů v obrazu) tak pokročilých algoritmů, které jsou schopny vložit grafické prvky do obrazu takovým způsobem, že divák zůstane na pochybách zda-li náhodou není daný prvek opravdu skutečně sníman kamerou nebo je čistě virtuální.

Tyto systémy současně přebírají data z objektivů kamer aby mohli grafiku upravit, adekvátně podle momentálního sklonu kamery a ohniska objektivu. Výsledky jejich práce je běžně možné vidět při přenosech atletiky nebo fotbalu. Vykreslují jak časomíry a klasickou grafiku tak virtuální lajny do videa (délka hodu, vzálenost od branky atd.)

Výhody:
- Pokročilé funkce umožnující rozsáhlou manipulaci s obrazem
- Bezkonkurenční manipulace z obrazem nedosažitelná jinou technologií
- Podpora externích datových zdrojů - zařízení může tedy zobrazovat libovolná data

Nevýhody:
- Přílišná komplexnost systému, vyžaduje 3-5 lidí obsluhu (nejmodernější systémy méně)
- Vysoká cena (stovky tisíc USD)
- Zařízení se zapojuje až za střižnu, což omezuje kontrolu stříhače a režiséra nad grafickou částí
- Spolupracuje pouze s nejdražšími televizními systémy, nemožné pro použití v levnější produkci

Příklady systémů:

1st & Ten - původně navrženo pro americký fotbal, společností SportVision která nyní nabízí celou řadu podobných systémů.

Systémy PVI Media Services - konkurence SportVision systémy jsou podobné.

Zhrnutí

Aplikace, která je cílem této práce je určena právě pro levnější produkci. Patří tedy do první kategorie, jelikož využívá klíčování a pro manipulaci s videem se využívá oddělené zařízení, tedy ne počítač.

[1] http://www.google.com/patents/US5917553?dq=5917553

3. Návrh aplikace

1. Použité technologie

Pro vývoj aplikace jsem použil .NET Framework od společnosti Microsoft. Samotná aplikace je psaná jako WPF v jazyce C#. Hlavním důvodem proč jsem zvolil právě architekturu .NET je fakt, že aplikace, využívá širokou škálu fukncí, které jsou v .NET Frameworku již implementovány, od tříd pro generování obrázků, serializaci binárních a xml, souborů, generátorů obrázků, práci s SQL, práci se Sockety, RS232 portem, Reflexe a načítání third-party Pluginů nebo například implentovaný Observer pattern v podobně C# events.

Důvodem proč jsem vybral právě technologii nejnovější technologii WPF je podopora DataBindingu. Tedy možnost dynamicky načítat data do uživatelských rozhraní s oboustrannou aktualizací dat. Právě Binding je v této aplikaci naprosto klíčový, kdy se data s editačního prostředí automaticky mění na živém výstupu.

Další výhodou WPF oproti WinForm je fakt, že uživatelské rohraní je vykreslováno grafickou kartou, což vede k mnohem plynulejšímu a rychlejšímu zobrazování prvků bez problikávání a podobných neduh WinForm.

2. Cílová platforma

Aplikace je navržena pro operační systém Windows. Byla vyvinuta a testována na Windowsu verze 8.1 s Frameworkem 4.5. Tato konfigurace je tedy současně zamýšlěna jako cílová platforma.

Aplikace může být používána, jak ve studiu na stolním PC nebo pracovní stanici, tak v přenosovém voze, na notebooku nebo zabudovaném počítači. Obvzláště při práci v přenosovém voze nejsou k dispozici velké obrazovky s vysokým rozlišením (kvůli nedostatku místa). Je tedy důležité optimalovat uživatelské rozhraní tak, aby byla aplikace přehledná a všechny její funkce viditelné i na menší obrazovce s nižším rozlišením.

3. Hardware

Aplikace stabilně využívá 1 vlákno, jsou-li spuštěny další funkce jako například timery, nebo načítání dat ze sítě či jiných portů, může být vláken spuštěných víc. Minimálně by cílový PC měl tedy obsahovat dvou-jádrový procesor, s hyperthreadingem nebo lepší.

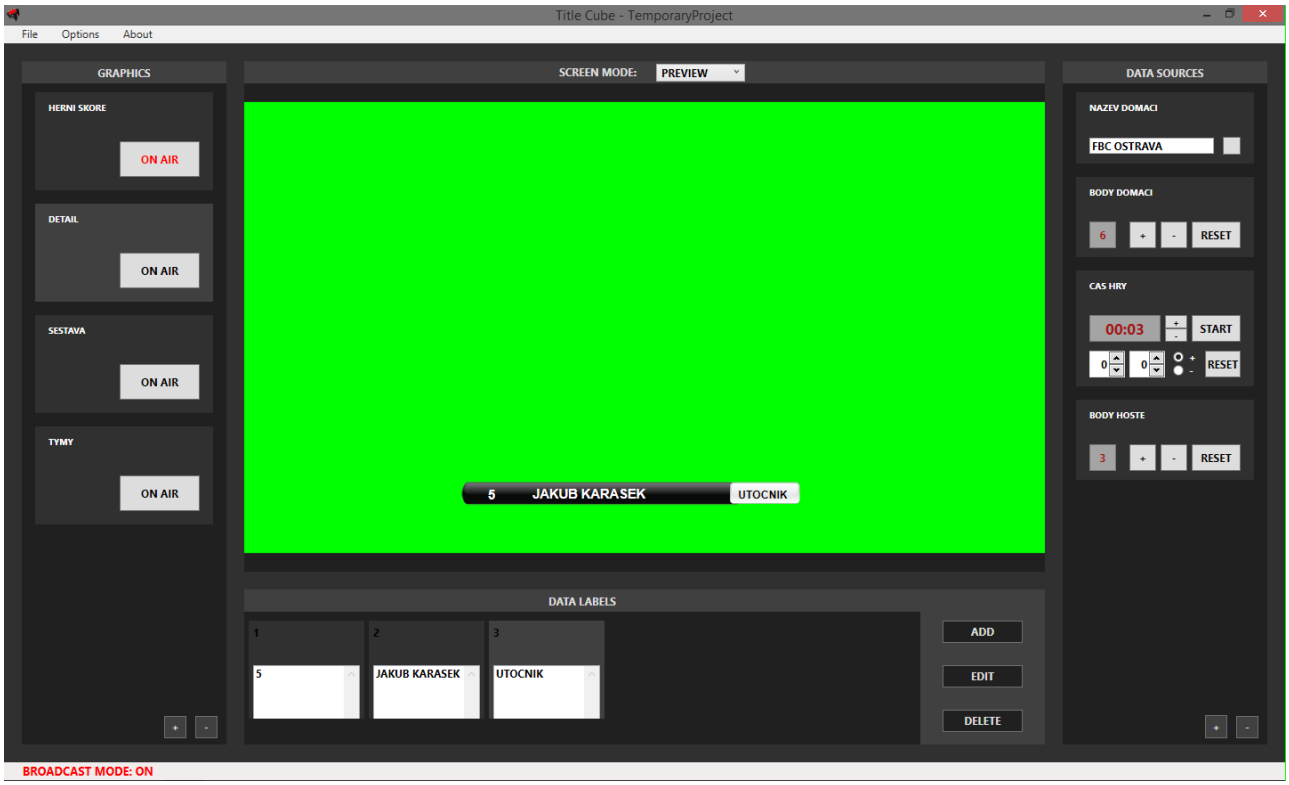
Nároky na pamět jsou závislé na počtu načtených grafických prvků a taky na použitém formátu obrázků. Je třeba si uvědomit že v nejhorším možném scénáři, grafika ve formátu Targa 32bit včetně alfakanálu, má ve fullHD rozlišení až 7.9MB, při načtení většího množství těchto obrázků může dojít ke značnému vytížení paměti.

Při použití zkomprimovaného formátu PNG se požadavky na pamět razantně sníží a celá aplikace by neměla vyžadovat více než desítky MB v RAM. Výsledku zatížení paměti a procesoru jsou v závěrečné kapitole Testování.

4. Požadavky na uživatelské rozhraní

Jak už bylo zmíněno v odstavci zabývajícím se cílovou platformou, pro uživatelské rozhraní je důležité aby umožňovalo rychlou a přehlednou práci i na malém displeji s nízkým rozlišením. Takovýmto rozlišením je myšleno dnes nejrošířenější širokoúhlé rozlišení počítačových diplejů, tedy 1366x768 pixelů. Uživatelské rozhraní samozřejmě musí být schopné využít canvasu monitoru s větším rozlišením, jelikož ne vždy bude aplikace využívána v terénu v útrobách přenosového vozu. Při práci ve studiu bude bude mít uživatel zcela jistě k dispozici větší zobrazovací plochu, ale zpravidla také náročnější scénář (více titulků, jako například při vysílání zpráv), je tedy důležité uživateli poskytnout maximální možné pohodlí a zobrazit co největší množství ovládacích prvků.

Při vývoji uživatelského rozhraní jsem vycházel ze zkušeností z praxe při práci s jinými titulkovacími editory. Současně jsem se řídil i odezvou od lidí, kteří tyto softwary obsluhují. Pro obsluhu je klíčové aby měla okamžity přehled o právě zobrazované grafice, otom kde je grafika v obraze a jaké data obsahuje. Dále obsluha vyžaduje mít přehled o dynamických datech bez překlikávání do podoken. To umožňuje rychlou kontrolu korektnosti dat a minimalizuje chyby v živé produkci.



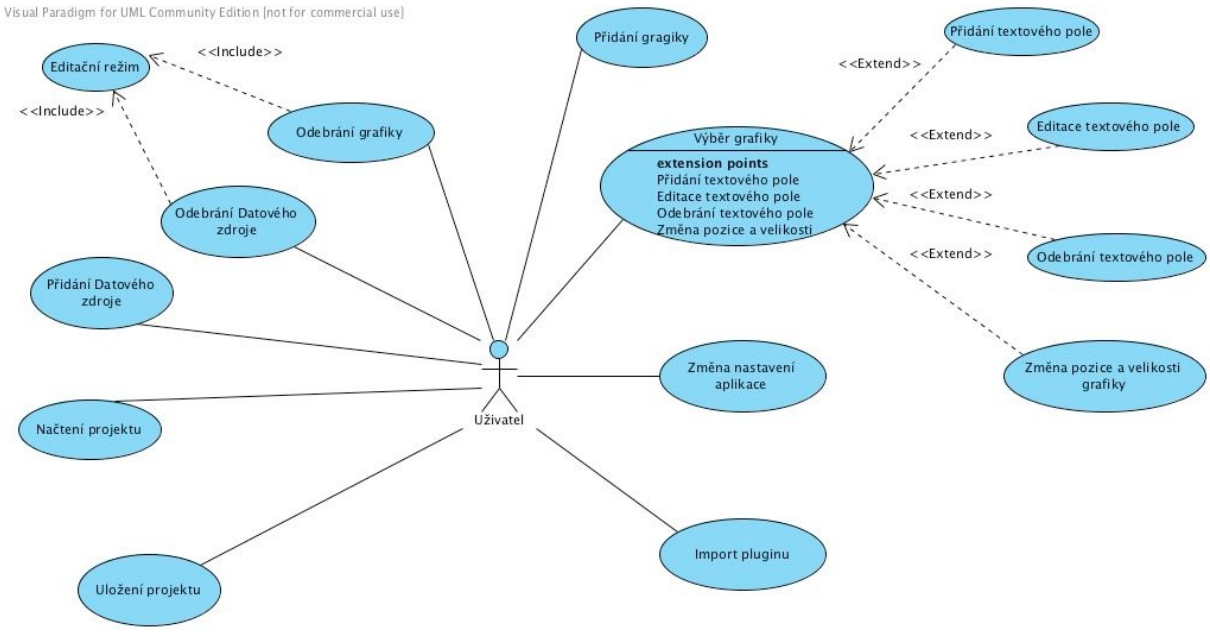
Obrázek 3.1: Hlavní okno

5. Základní funkce programu

Aplikace musí být schopna vytvářet grafické prvky, tedy kontejnery s obrázkem jako pozadí, do kterého budeme moci dynamicky vpisovat data. Dále tedy potřebujeme mít možnost umístit data do grafického prvku. K tomu slouží prvek označovaný jako datové pole (DataLabel), který je plně editovatelný (jeho poloha, velikost, písmo…). Tento prvek může sloužit jako přímy textový vstup nebo jako zobrazovací plocha pro datový zdroj.

Dále se tedy dostáváme k datovým zdrojům. Jedná se o logické prvky programu, které poskytují textové hodnoty pro datová pole. Dále je důležitá možnost rožšiření programu o vlastní datové zdroje pomocí pluginu.

V poslední řadě je důležitou funkcí aplikace schopnost uložit veškerá nastavení do souboru a ten později znova načíst. Výroba profilu trvá poměrně dlouho je tedy důležité nezdržovat obsluhu nastavováním programu pro konkrétní přenos (například florbalu), ale mít už hotový projekt, který jen čeká na využití.



Obrázek 3.2: UseCase Diagram

4. Zapojení do systému

1. Konexe

Aplikace musí mít možnost a být schopna posílat signál, jak na výstup grafické karty, tak na výstup speciálních karet. V televizních systémech se již dnes používají výhradně digitální linky pro vedení obrazu, důvodem je vyší kvalita obrazu, jeho odolnost vůči zarušení elektromagnetickým polem a konstantní kvalitou. Je tedy důležité, aby systém byl schopný poskytnout digitální signál buď v podobě HDMI nebo profesionály více oblíbeného SDI (resp. jeho variant SD-SDI, HD-SDI, 3G SDI, 6G SDI). SDI výstup se dá řešit například HDMI / SDI převodníkem nebo případně intergrovanou SDI kartou v PC.

Obrázek 4.1: Ukázky konektorů

2. Výstupní rozlišení

Všechny v současné době používané video normy musí být nastavitelné, je třeba zvolit jiné výstupní rozlišení pro klasický SD signál (720x576), 16:9 SD signál (1024x576), HD signál (1280x720) a FullHD signál (1920x1080). Do budoucna se počítá i s rozšířením na UltraHD kde výstupní rastr odpovídá 3840x2160. V současné době ještě nejsou ustálené standardy pro přenos 4K videa, nemá cenu se tedy tímto rozlišením zabývat. Samotné nastavení normy ovšem není věc aplikace, ale ovladače grafické karty.

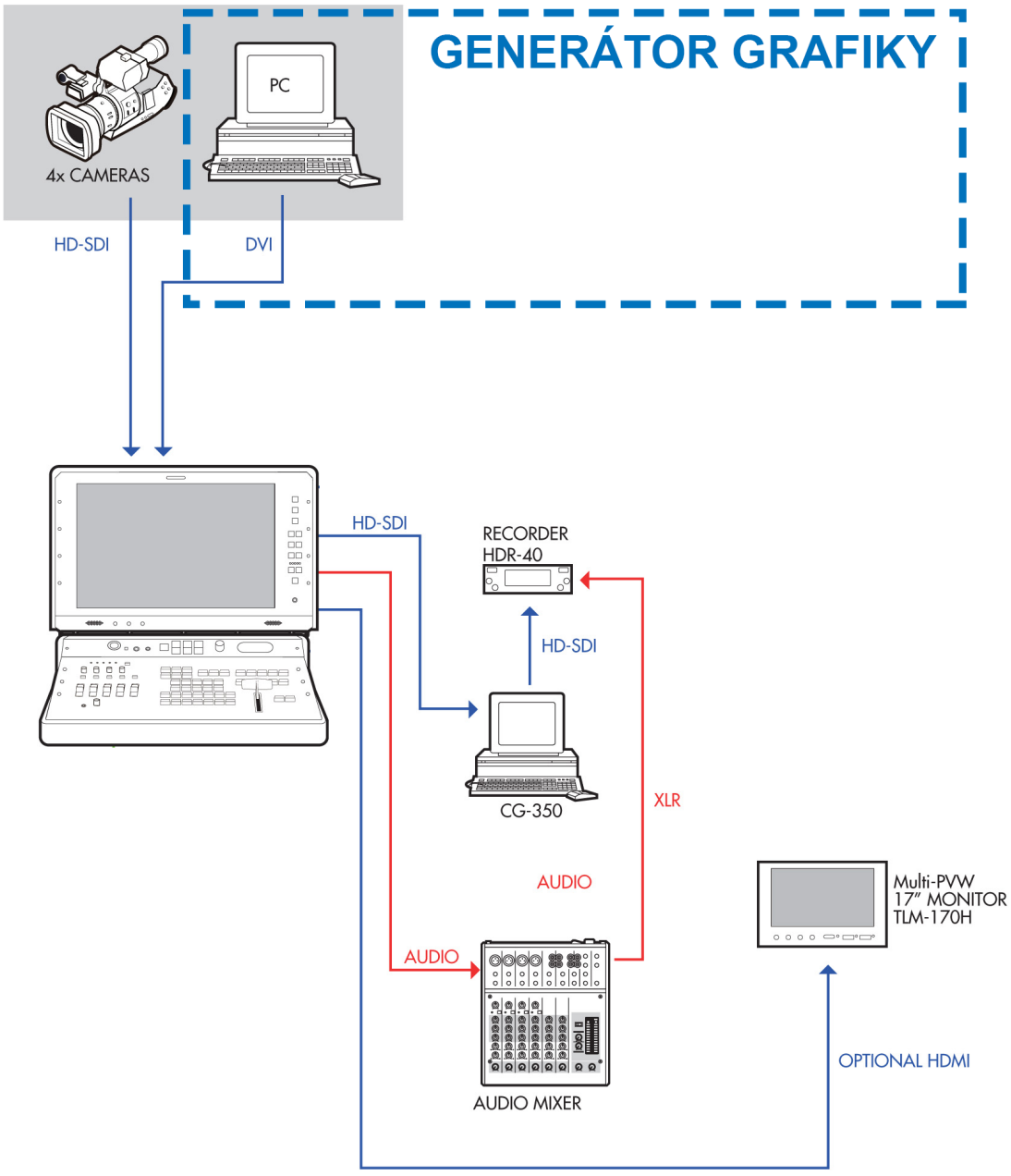
3. Snímkování

Nastavení snímkovacích frekcencí výstupního signálu, je velmi důležitým faktorem, který může ovlivnit například plynulost animací v motion grafice. Pro zajistění správné výstupní normy je potřeba znát v jakém standardu probíhá video produkce (například 720p50 - 720 řádků a 50 progresivních snímků za vteřinu) a podle toho nastavit grafický adaptér.

4. Workflow

Neboli způsob zapojení celého systému ukazuje v jaké části televizního systému se aplikace nachází. Pro jednoduchost jsem schéma zbavil přebydků jako video servery, audio, vnitřní komunikace. Jde především o čisté propojení video signálů.

Na schématu je důležité si uvědomit, že zdroj grafiky je přiveden do střižny současně s ostatními video signály (např. kamery). To umožnuje stříhači částečnou kontrolu na zobrazovanou grafikou. Současně je takovéto zapojení bezpečnější. V případě selhání generátoru obrazu nedojde k výpadku celé produkce. U overlay systému jde z pravidla o zapojení grafické jednotky až za výstup ze střižny, což zbavuje střihače kontroly nad grafikou.



Obrázek 4.2: Workflow

5. Implementace

Aplikace je rozdělená do několika základních částí. Hlavní částí jsou uživatelská rozhraní, které slouží jako řídící jednotka celé aplikace. Dále jsou to třídy jádra aplikace, jedná se o třídy držící v kolekcích data o prvcích které jsou později zobrazovány v uživatelském rozhraní a na výstupní obrazovce.

Dalším důležitou částí aplikace je abstraktní třída DataSource, která slouží pro polymofyzmus dalšími třídami, které reprezentují logiku jednotlivých datových zdrojů. Třída se nachází v odděleném namespace, aby ji bylo možno šířit, pro případné developery externích datových zdrojů.

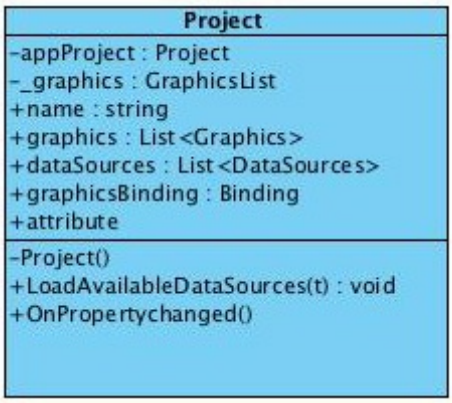
Interní datové zdroje obsahují základní funkce potřebné pro živou produkci, jako ukládání textu, počítání bodů, a časomíra. Program je možno nadále rozšiřovat o prakticky neomezené množšství externích datových zdrojů pro specifické účely. Nyní se podíváme na klíčové třídy jednotlivých namespaců podrobněji. Uživatelské rozhraní rozebereme v další kapitole.

1. Namespace ControlApplication.Core

Tento namespace obsahuje třídy klíčové pro běh aplikace, třídy držící data o jednotlivých prvcích. Současně ovšem i třídy sloužící k editaci prvků v uživatelském rozhraní.

Třída Project

Implementovaná pomocí vzoru Singleton, stejná instance je tedy k dispozici napříč celou aplikací. Obsahuje především ObservableCollections jako graphics a dataSources. Současně obsahuje název projektu a kolekci udržující datové typy dostupných datových zdrojů.



Obrázek 5.1: Třída Project

Třídy GraphicsList, DataSourceList, DataLabelList

Jedná se o třídy typu ObservableCollection<T>. Jde o kolekci, která nashlásí naslouchajícímu prvku uživatelského rozhraní změnu a umožní automatickou aktualizaci pomocí Bindingu.

ObservableCollection<T> Class
"Represents a dynamic data collection that provides notifications when items get added, removed, or when the whole list is refreshed."[2]

```
// Ukázka implementace ObservableCollection<T>
public class GraphicsList : ObservableCollection<Graphics>
{
    public GraphicsList()
    {

    }
}
```

Tyto kolekce drží objekty tříd Graphics, DataLabel, DataSource. Tedy třídy, které udržují číselné data reprezentující jednotlivé prvky. Pojdme si je projít podrobněji.

Třída Graphics

Je zřejmé, že tato třída obsahuje všechny informace týkající se grafického prvku, krom názvu, podkladového obrázku, velikosti a polohy obsahuje ObservableCollection se všemi datovými poli. Zajimavostí je, že třída implementuje rozhraní INotifyPropertyChanged[3]. Toto rozhraní umožnuje implementaci akutalizačního eventu PropertyChangedEventHandler[4]. Současně se vytvoří i patřičný handler který se volá s názvem dané property při aktualizaci.

```
protected void OnPropertyChanged(string name)
{
        PropertyChangedEventHandler handler = PropertyChanged;
        if (handler != null)
        {
           handler(this, new PropertyChangedEventArgs(name));
        }
}
```

---

[2] http://msdn.microsoft.com/en-us/library/ms668604(v=vs.110).aspx

[3] http://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged(v=vs.110).aspx

[4] http://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged.propertychanged(v=vs.110).aspx

Ukázka implementace proměnné name pro následný Data Binding[5].

```
// privátní proměnná
private string _name;

public string name
{
        get
        {
                return _name;
        }
        set
        {
                _name = value;
                OnPropertyChange(“name”); // Zavolání eventu
        }
}
```

Při každé změně property se zavolá event který aktualizuje všechny naslouchající prvky. Nastavení Bindingu pro TextBlock[6] v XAML se provádí takto:

```
<TextBlock Text="{Binding name}”/>
```

Přičemž se rodičovské okno tohoto elementu musí mít nastavenou property DataContext[7] na objekt ve kterém se nachází property name. To řekne kompilátoru kde má danou property hledat. Pokud bychom chtěli vytvořit Binding z CodeBehind v C# je postup lehce složitější.

```
Binding nameBinding = new Binding("name");
nameBinding.Mode = BindingMode.TwoWay;
nameBinding.Source = this;

textBlock.SetBinding(TextBlock.TextProperty, nameBinding);
```

Podobně je Binding implementovám i ve třídě DataLabel na většinu properties.

---

[5] http://msdn.microsoft.com/en-us/library/ms752347(v=vs.110).aspx

[6] http://msdn.microsoft.com/en-us/library/ms754356(v=vs.110).aspx

[7] http://msdn.microsoft.com/en-us/library/system.windows.frameworkelement.datacontext(v=vs.110).aspx

Třída DataLabels

Reprezentuje pole pro zobrazení dat v grafice, třída udržuje informace o pozici pole, jeho velikosti a formátu textu, tedy všechny hodnoty o typu písma, velikosti, zarovnání a barvě. Dále obsahuje možnost vložení obrázku, a přiřazení datového zdroje pro naslouchání.

Třída Settings

Obsahuje základní nastevní aplikace a metody pro uložení konfiguračního nastavení. Základními vlastnostmi třídy jsou informace o výstupní normě generovaného signálu, barvě pozadí a posledním použitém projektu pro opětovné načtení.

Třída ProjectController

Obsahuje metody OpenProject a SaveProject pro ukládání a načítání celé třídy Project do XML souboru. Pro práci s XML je využita třída XmlDocument[8]. Třída obdrží název složky ve kterém se nachází soubor project.xml . Složka s projektem nadále obsahuje všechny obrázkové podklady ve složce img.

Ukázka výsledného XML

```
<project name="TemporaryProject">
 <dataSourceList>
  <dataSource id="0" value="FBC OSTRAVA" name="NAZEV DOMACI" type="DataSources.TextHolder" />
   <dataSource id="1" value="6" name="BODY DOMACI" type="DataSources.PointCounter" />
   <dataSource id="2" value="00:00" name="CAS HRY" type="DataSources.GameTimer" />
   <dataSource id="3" value="3" name="BODY HOSTE" type="DataSources.PointCounter" />
 </dataSourceList>
 <graphicsList>
   <graphics id="0" name="HERNI SKORE" left="106" top="20" width="162" height="21"
background="new_game.png">
     <dataLabel id="0" data="0" dataSource="1" left="9" top="5" width="11" height="13" background="" />
    <dataLabel id="5" data="0" dataSource="3" left="146" top="5" width="10" height="16"
background="" />
   </graphics>
   <graphics id="1" name="DETAIL" left="200" top="349" width="311" height="21"
background="new_title.png">
     <dataLabel id="1" data="5" dataSource="-1" left="24" top="5" width="13" height="13"
background="" />
     <dataLabel id="2" data="JAKUB KARASEK" dataSource="-1" left="65" top="4" width="121"
height="13" background="" />
    <dataLabel id="3" data="UTOCNIK" dataSource="-1" left="243" top="4" width="68" height="14"
background="" />
   </graphics>
   <graphics id="2" name="SESTAVA" left="196" top="39" width="349" height="324"
background="new_lineup.png">
     <dataLabel id="4" data="TYM DOMACICH" dataSource="-1" left="127" top="3" width="107"
height="15" background="" />
   </graphics>
   <graphics id="3" name="TYMY" left="205" top="313" width="307" height="63"
background="new_teams.png" />
 </graphicsList>
</project>
```

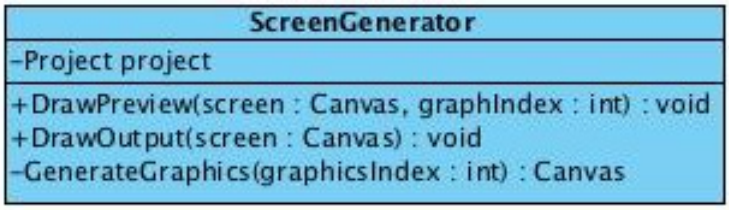
[8] http://msdn.microsoft.com/en-us/library/system.xml.xmldocument(v=vs.110).aspx

Třída ResizingAdorner : Adorner[9]

Tato třída obsluhuje zvětšování a zmenšovaní kontrolních prvků za běhu aplikace. V kombinaci s DataBindingem je tedy možné rychle a hlavně za běhu měnit vlastnosti grafických prvků ve vysílání. Například zvětšování, zmenšování, změna pozice atd. Jedná se o veřejně dostupnou třídu řešící tyto operace.

Třída ScreenGenerator

Obsahuje mimo jiné metodu GenerateGraphics pro vykreslení grafického prvku. Metoda vrací UIElement[10] Canvas[11] který reprezentuje jak v náhledu tak na výstupní obrazovce grafický prvek. Metoda současně obsluhuje náhledovou obrazovku stejně jako výstupní obrazovku.



Obrázek 5.2: Třída ScreenGenerator

9 http://msdn.microsoft.com/en-us/library/vstudio/ms771714(v=vs.90).aspx

10 http://msdn.microsoft.com/en-us/library/system.windows.uielement(v=vs.110).aspx

11 http://msdn.microsoft.com/en-us/library/system.windows.controls.canvas(v=vs.110).aspx

2. Namespace AbstractDataSource

Třída DataSource
Abstraktní třída reprezentující všechny typy datových zdrojů, od textových vstupů, přes timery, až po specifické externerní pluginy jako například počítadlo bodů pro stolní tenis. Třída obsahuje property name pro rozeznání jednotlivého DataSource uživatelem, property ID pro unikátní identifikaci objektu, dále obsahuje property string value, poskytující aktuální data z datového zdroje v podobě řetězce.

Třída se nachází v externí assembly z důvodu nutnosti její implementace do rozšiřujících pluginů. Třetí strana vyvíjející plugin si načte abstraktní třídu DataSource nastaví ji jako rodiče své třídy reprezentující plugin. Aplikace si pak při náčítání assembly třetí strany ověří pomocí Reflection zda daná třída dědí právě ze třídy DataSource. Při úspěšném ověření pak lze použít dodanou třídu pro polymorfizmus.

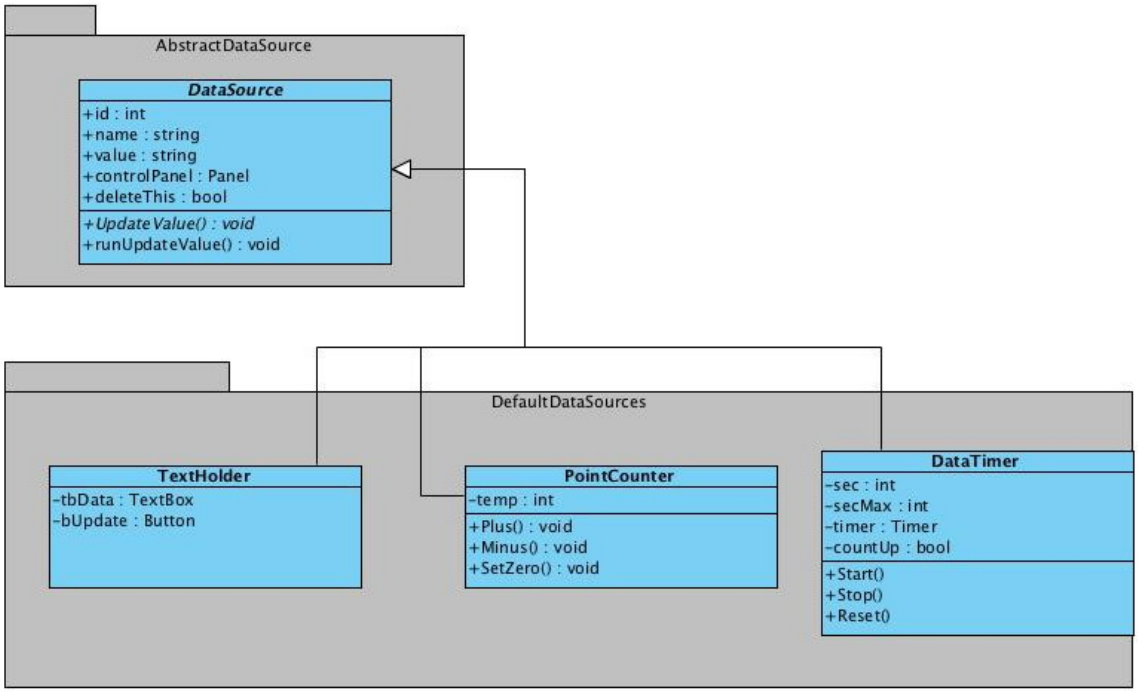
3. Namespace DataSources

Obsahuje třídy reprezentující defaulutní DataSources řešící základní problematiku živých přenosů. Jednotné texty ve více grafických prvcích, například název soutěže nebo vysílané události. Jednoduché přepínání bodových stavů. Časy a odpočty ve videu.

Třída TextHolder
Datový zdroj TextHolder funguje jako jednoduchá pamět umožnující uživateli, uložit libovolnou textovou hodnotu do paměti, a následně ji vyvolat ve více grafikckých prvcích. Uživatelské rozhraní se skládá z textboxu umožňující zadávání dat. Tlačítka pro update dat.

Třída PointCounter
PointCounter slouží k jednoduchému přičítání bodových stavů, bez nutnosti mačkat klávesnici, aktualizace dat probíhá hned po změně hodnoty.

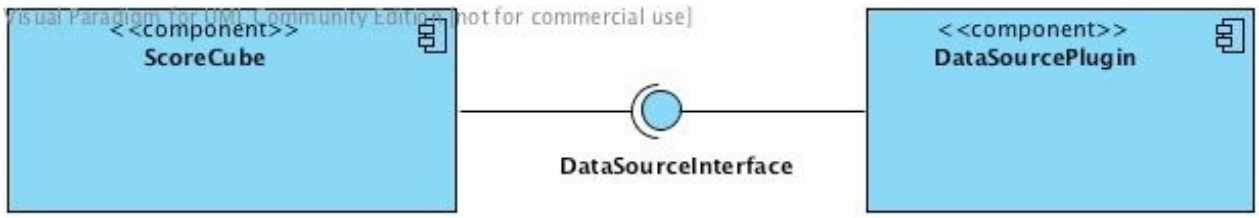
Třída GameTimer
Tato třída obsahuje vlastní nezávislý timer běžící na vlastním vláknu. Lze nastavit vzestupné nebo sestupné počítání času, startovní a cílový čas. Dále obsahuje metody pro korekci časových hodnot. Slouží k zobrazování časomír, zejména ve sportovních přenosech.



Obrázek 5.3: Diagram Tříd - DataSource

4. Namespace PluginDataSource

Tento namespace je rezerován pro vývoj dodatečných pluginů. Třetí strany si mohou samy doprogramovat vlastní datasource včetně uživatelského rozhraní. Potřebuje-li uživatel specifický datasource pro svůj účel například, logicky vyřešené počítadlo pro tenis s ergonomickým ovládáním (tlačítky pro 0, 15, 30, 40, A). Má možnost si takovýto plugin dodělat dle dokumentace. Dalšími případy specialních datasource můžou být například Socket client, který načítá data z daného TCP či UDP portu, nebo případně datasource pro práci s RS232 který se hojně používá pro řízení cedulí na sportovních halách. Je tedy možné software přímo připojit na tuto ceduli a nezetěžovat obsluhu opisováním dat.



Obrázek 5.4: Diagram Komponent Pluginu

6. Uživatelské rozhraní

Při návrhu uživatelského rozhraní byl kladen velký důraz na ovladatelnost aplikace na menších obrazovkách s nižším rozlišením, které se často nacházejí v přenosových vozech a mobilních režijních stanovištích. Důležitým prvkem je to, aby obsluha měla stály přehled nad výstupem, který posílá do vysílání, a současně měla možnost rychle modifikovat data grafiky bez nutnosti skrýt grafiku v obraze.

1. Hlavní menu

- File
    - New - vytvoří nový prázdný projekt
    - Open - načte existující projekt ze souboru
    - Save - uloží projekt do existujícího souboru
    - Save As - uloží projekt pod novým názvem do nového souboru
    - Quit - ukončí aplikaci

- Options
    - Broadcast mode
        - On - zapne grafický výstup pro vysílání
        - Off - vypne grafický výstup pro vysílání
    - PluginImport - zobrazí formulář PluginImport
    - Settings zobrazí okno ApplicationSettings
- About
    - Version - Zobrazí informace o programu
    - Help - Zobrazí nápovědu

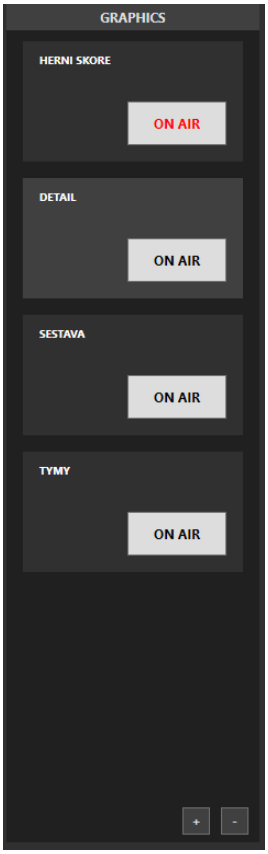
2. Sektor Grafika (Graphics)

Nachází se v levé části hlavního okna. Tato sekce je určena ke stružení všech jednotlivých grafických prvků. Jednotlivé grafické prvky jsou reprezentovány panelem, který obsahuje:
- Titulek grafického prvku
- Tlačítko OnAir
- Tlačítko pro smazání grafického prvku

Celý panel současně funguje jako tlačítko. Kliknutí na panel daného grafického prvku způsobí že id daného prvku se uloží jako do proměnné selectedGraphics, která říká který prvek je určen k editaci (změna velikosti, pozice, přidání datových polí, atd.). Toto současně způsobí zobrazení daného grafického prvku na náhledové obrazovce a načtení příslušných datových polí. Editovat lze vždy jen jeden prvek.

OnAir tlačítko slouží k označení grafiky, která jde přímo na výstupní obrazovku, nebo na náhledovou obrazovku je li v režimu zobrazení výstupu.
Skryté tlačítko pro smazání a editaci daného prvku je k dipozici pouze tehdy je-li celý panel Graphics v editačním režimu. Tento režim lze zapnout i vypnout tlačítkem se znakem “ - “ nacházejícího se v dolní části panelu. Vedle něj se nachází tllačítko se znakem “ + “ určené k přidání nového grafického prvku. Po jeho zmáčnutí se zobrazí formulář GraphicsDetail.

Pro uložení všech ovládacích panelů grafických prvku jsem použíl prvky StackPanel a ItemsControl, který se narozdíl od běžného Canvas sám stará o řazení nových prvků a to buď ve svislé nebo vodorovné ose. Dále umožňuje přidání posuvníku, překročí-li obsah jeho rozměry.



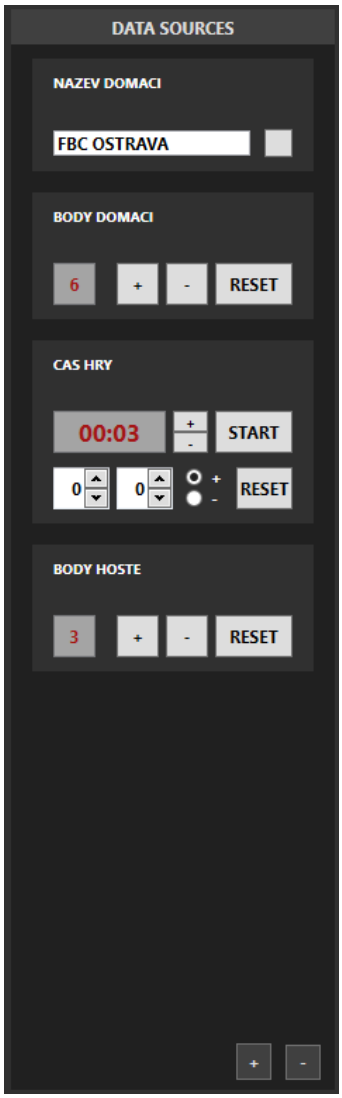
Obrázek 6.1: Sekce grafických prvků

3. Sektor Datové zdroje

Sektor datových zdrojů je řešen podobně jako sektor grafických prvků. Základem je tedy ItemsControl, který drží kolekci jednotlivých panelů, každý reprezentující jeden datový zdroj. Ovládací panely datových zdrojů se liší podle typu datového zdroje, jelikož každý má jiné nároky na nastavení funkce. Datový typ TextHolder má pouze vstupní pole na uložení textu a tlačítko pro aktualizaci hodnot. Oproti tomu například DataTimer obsahuje vlastní podformulář s detailním nastavením timeru.

Vzhled a funkčnost ovládacího panelu je zapouzdřen ve třídě reprezentující datový zdroj, hlavní formulář si tedy pouze načítá property control z daného objektu třídy DataSource. Toto řešení bylo zvoleno hlavně proto, aby si vývojáři dodatečných pluginů mohli navrhnout i kontrolní panel.

Součásti je stejně jako u sektoru grafických prvků tlačítko pro editační režim, které po kliknutí zobrazí na všech kontrolních panelech tlačítko pro odstranění panelu a nechybí ani tlačítko pro přidání nového datového zdroje.



Obrázek 6.2: Sekce datových zdrojů

4. Sektor Datová pole

V dolní části hlavního okna pod náhledovou obrazovkou se nachází panel pro datová pole. Datová pole jsou zobrazovací plochy pro data v grafice. Každý grafický prvek má tedy svou kolekci datových polí.
V programu rozlišujeme datová pole třech typů:

- Textové
- Datový zdroj
- Obrázek

Textová pole
Umožnují přímou editaci textu v grafice bez nutnosti jakékoliv další operace, text se přepisuje přímo během psaní uživatele a to jak v náhledu tak ve výstupním signálu (je-li daná grafika ve stavu OnAir).

Datový zdroj
Datové pole se prováže s příslušným datovým zdrojem, ten pak při aktualizaci změní text v datovém poli. Jeden datový zdroj může mít teoreticky neomezené množství připojených datových polí, využívá se Binding na property value;

Obrázek
Slouží k vložení obrázku nad podkladovou grafiku. Vhodné například pro fotky hráčů, loga klubů, soutěží nebo sponzorů. Obrázky jsou načítány ze souboru, po uložení projektu jsou uloženy do složky v projektovém adresáři.

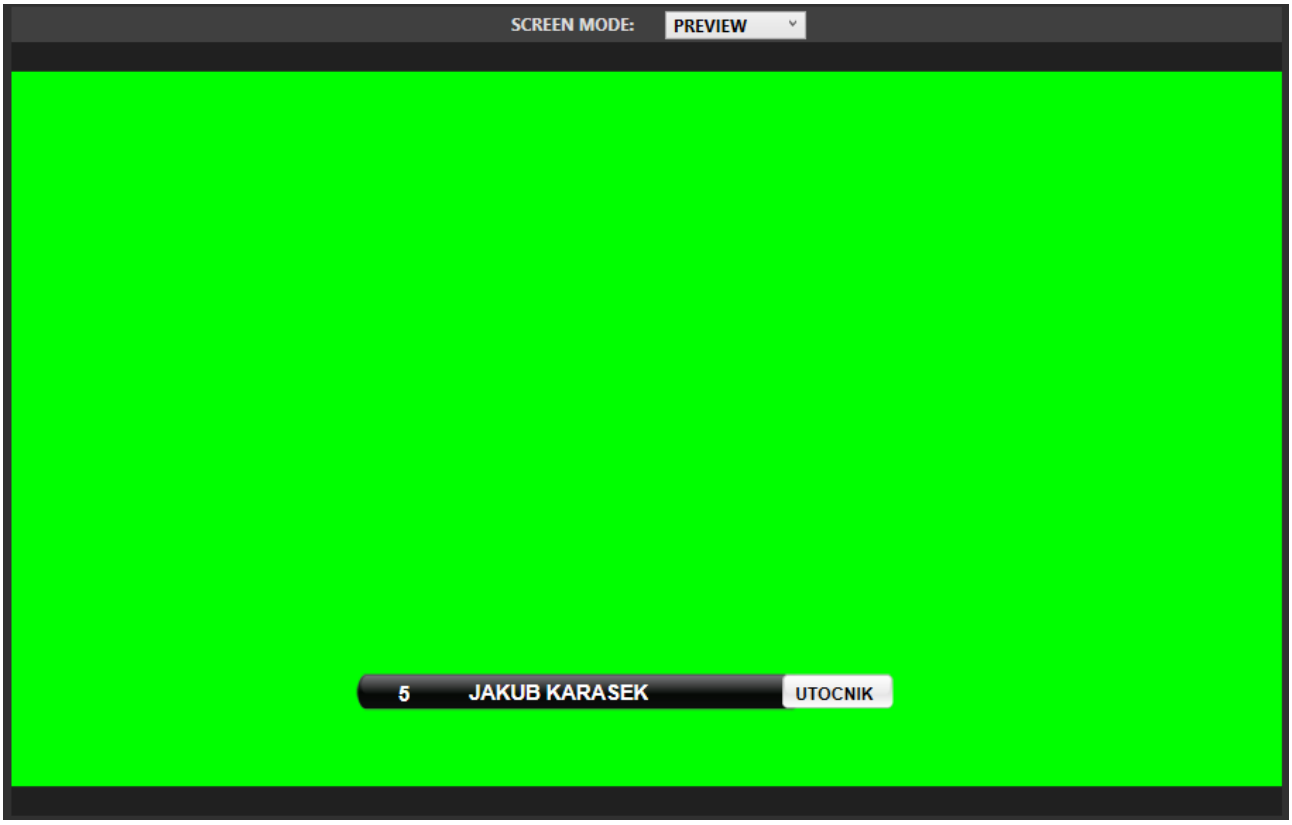
Obrázek 6.3: Sekce datových polí

5. Náhledová obrazovka

Patří k nejdůležitějším nástrojům aplikace. Obsluha může s její pomocí měnit velikost i pozici grafikých prvků. Má možnost upravovat pozici datových polí, ale hlavně může připravovat novou grafiku a současně do vysílání posílat grafiku jinou. Náhledová obrazova pracuje ve dvou režimech.

Režim náhledu (Preview)
Umožňuje editaci kteréhokoliv grafického prvku, nezávisle na vysílání. Výběr grafiky pro editaci se provádí v sektoru Grafických prvků. Při kliknutí na panel prvku se daná grafika zobrazí na náhledové obrazovce. Současně s ní se zobrazí všechny datové pole s jejich hodnotami.

Režim vysílání (OnAir)
Slouží ke kontrole vysíláného obsahu, vhodný hlavně je-li vysílán více než jeden grafický prvek současně. Jelikož tento režim zobrazuje přesnou kopii výstupního signálu, je možné v kontrolovat například kompozici obou prvků, jestli k sobě jak se říká “ladí”. Současně může nastat situace, kdy obsluha z provozních důvodů nemá možnost sledovat programový monitor (monitor zobrazující výsledný postříhaný video signál z režie). Pak, je tento režim náhledové obrazovky jedinou možností, jak může obsluha vidět jakou grafiku opravdu posílá do režie.



Obrázek 6.4: Editační obrazovka

6. Stavový řádek

Zobrazuje textové informace o aktualním stavu aplikace. Nachází se na dolní hraně hlavního okna aplikace. Ve většině případů zobrazuje stav vysílání. Je-li vysílací obrazovka odpojena - přepínáč Broadcast nastaven na OFF, znamená to, že okno zobrazující výstupní signál je zavřeno. V takovémto případě program není závislý na připojení externí obrazovky a aplikace může sloužit k přípravě vysílání a editaci profilů.

V této situaci stavový řádek hlásí zprávu:
BROADCAST: OFF

Při přpnutí přepínače Broadcast do polohy ON si nejdříve aplikace ověří dostupnost externího monitoru a jeho správné nastavení, poté spustí třídu pro generování signálu a spustí samotný proces generování výstupního obrazu.

V tuto chvíli staový řádek hlasí zprávu:
BROADCAST: ON

Obrázek 6.5: Stavový řádek

7. Generátor výstupu

O zobrazení výstupního signálů se stará okno Output. Okno si automaticky zjistí šířku primárního monitoru a posune se nalevo od něj na pozici sekundárního monitoru. Poté se okno zvětší na fullScreen a zabere celou sekundární obrazovku.

Ačkoliv se může řešení zdát kostrbaté, WPF zatím neumí čistě pracovat s dvěmi obrazovkami. Je tedy důležité hledání alternativních řešení jak okno na druhou obrazovku dostat.

Při načtení a zvětšení okna může začít samotné generování výstupu. To velikosti prvků uložené v proměnných se ovšem vztahují pouze na obrazovku náhledovou. Pro zvětšení všech prvků v poměru stran se využije prvek WPF zvaný Viewbox[12]

```
//Ukázka XAML kódu pro okno Output
<Viewbox HorizontalAlignment="Stretch" VerticalAlignment="Stretch" Stretch="Uniform">
      <Canvas x:Name="cOutput" Background="Lime" Width="736" Height="414">

      </Canvas>
</Viewbox>
```

Jelikož obě property jak HorizontalAlignment tak VerticalAlignment jsou nastaveny na hodnotu Strech, roztáhnou se na plné rozlišení okna (sekundárního displeje). Hodnota Uniform znamená, že vždy bude dodržen původní rozměr (v našem případě 16:9).

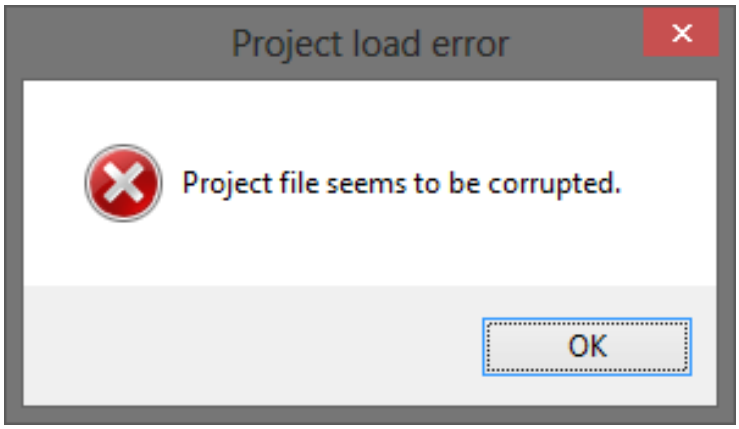
Canvas cOutput slouží již čistě jako kontejner obsahující zobrazované grafické prvky, jeho startovní velikost je 736, 414px. Tedy stejná jako referenční hodnota náhledové obrazovky. Stejné počáteční hodnoty zaručí, obě obrazovky budou zvětšovat úměrně velikosti obrazovky a všechny poměry budou stejné.

[12] http://msdn.microsoft.com/en-us/library/system.windows.controls.viewbox(v=vs.110).aspx

8. Chybové zprávy

1. Chyba načítaní projektu

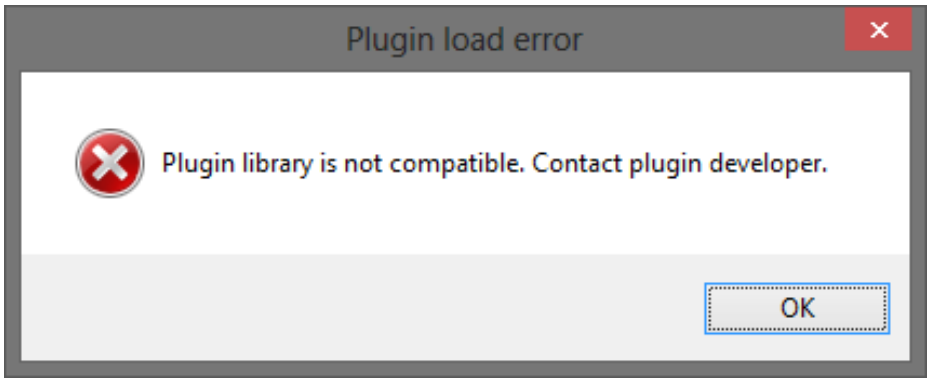
Chyba se zobrazí je-li problém při načtení uloženého projektu. Problém může být způsoben potencionálním poškezením souboru nebo jeho nekorektním uložením.



Obrázek 8.2: Chybová hláška - Project Load Error

2. Chyba načtení pluginu

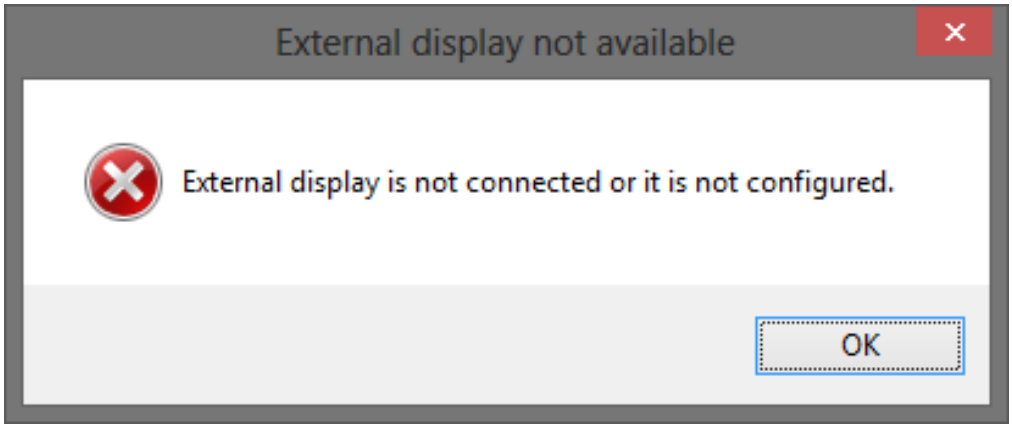
Oznamuje že načtení pluginu se nevydařilo. Může tak být v důsledku nekompatibilní knihovny. Nebo taky špatně implementované knihovny.



Obrázek 8.2: Chybová hláška - Plugin Load Error

3. Externí výstup není připojen nebo nakonfigurován

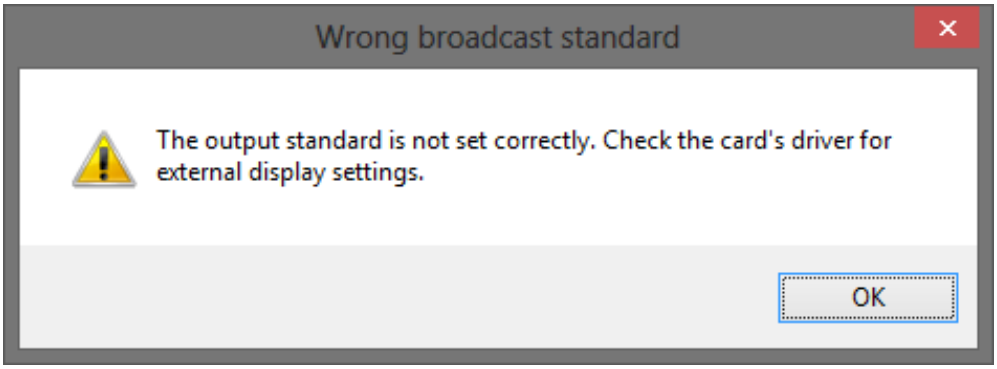
Program nerozpoznal žádný připojený externí výstup. Kabel není zapojen nebo není dobře nakonfigurovaná grafická karta.



Obrázek 8.3: Chybová hláška - Display Not Available

4. Externí výstup je špatně nakonfigurován

Applikace rozpoznala externí monitor ovšem nastavení výstupního formátu nesouhlasí s nastavením aplikace. Je potřeba změnit nastavení grafické karty aby odpovídalo nastavení aplikace, nebo změnit nastavení aplikace.



Obrázek 8.4: Chybová hláška - Wrong Broadcast Standard

9. Testování

Při testování aplikace jsem došel k zjištění, náročnost aplikace velmi kolísá v závislosti na použitých zdrojích. Záleží na velikosti podkladových obrázků i na jejich kompresi, dále záleží na počtu spuštěných datových zdrojů a počtu grafických prvků v projektu.

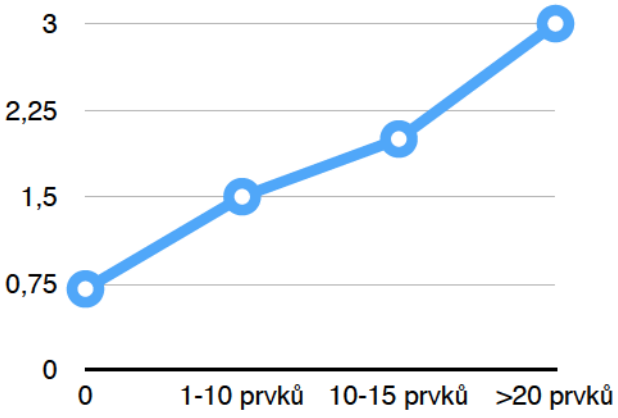
Testovací sestava

MacBook Air13" 2012
QuadCore i7 2.0Ghz
8GB Ram DDR3
256GB SSD

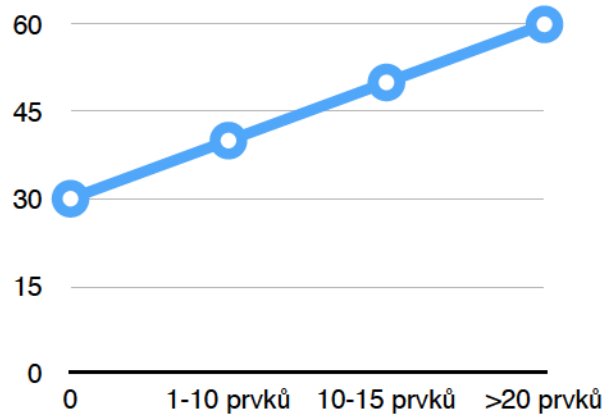
Operační systém - Windows 8.1 (nevirtualizovaný !)
Framework v. 4.5

Metodika

Zdrojové obrázky byly ve formátu PNG, jejich velikost odpovídala FullHD rozlišení. Výstup aplikace byl taktéž nastaven na FullHD.



Obrázek 9.1: CPU



Obrázek 9.2: RAM

10. Závěr

Aplikace kterou jsem nazval pracovním názvem TitleCube splňuje všechny body, které jsou klíčové pro úspěšné odbavení živého televizního přenosu. Je schopná v realném čase vytvářet videostream na klíčovatelném pozadí, který je spracovatelný bežným televizním hardwarem. K řešení není potřeba speciálních, výkonných a velmi drahých obrazových procesorů, jelikož aplikace pracuje na bázi klíčování ne overlay.

Aplikace postavené na podobných principech byly již v praxi testovány, například při přenosech Mistrovství světa v házené žen 2013 v Ostravě Porubě, ve stejném roce s nimi bylo odbavováno playoff extraligy můžu ve stolním tenise, a další desítky sportovních pořadů, které lze najít v archivu České televize. Dále v dnešním roce se s aplikacemi odbavovaly například online streamy florbalové extraligy a basketbalové NBL. Jedná se tedy o praxí osvědčený systém, který je hlavně svou dostupnou cenou zajimavý pro menší televizní studia s omezeným rozpočtem, ale kvalitou dokáže konkurovat a v mnoha funkcích i překonat drahé profesionální televizní řešení.

TitleCube je dalším skokem ve vývoji těchto aplikací které poskytují klíčovatelný obsah, narozdíl od původních aplikací umožnuje editaci a vytvoření vlastních profilů dle potřeby (původní aplikace byly psány na míru konkrétnímu sportu).

Technologie WPF nabízí obrovský potenciál pro další rozšiřování této aplikace o pokročilejší fuknce jako je motion graphics nebo animace. Aplikace určitě do budoucna nabízí způsob jak zaplnit díru na trhu způsobenou nedostatkem právě těchto aplikací, umožnující generovat klíčovatelný obsah v realném čase na grafický výstup.

11. Použitá literatura

[1] Jay Glynn a kol.: C# Programujeme profesionálně, COMPUTER PRESS, ISBN: 9788025100851

[2] Wilkinson, B., Allen, M.: Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers. Prentice Hall, 1999, ISBN: 0-13-671710-1

[3] Jim Owens.: Television Sports Production, FOCAL PRESS, ISBN: 9780240809168

12. Internetové zdroje

[1] http://www.google.com/patents/US5917553?dq=5917553

[2] http://msdn.microsoft.com/en-us/library/ms668604(v=vs.110).aspx

[3] http://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged(v=vs.110).aspx

[4] http://msdn.microsoft.com/en-us/library/system.componentmodel.inotifypropertychanged.propertychanged(v=vs.110).aspx

[5] http://msdn.microsoft.com/en-us/library/ms752347(v=vs.110).aspx

[6] http://msdn.microsoft.com/en-us/library/ms754356(v=vs.110).aspx

[7] http://msdn.microsoft.com/en-us/library/system.windows.frameworkelement.datacontext(v=vs.110).aspx

[8] http://msdn.microsoft.com/en-us/library/system.xml.xmldocument(v=vs.110).aspx

[9] http://msdn.microsoft.com/en-us/library/vstudio/ms771714(v=vs.90).aspx

[10] http://msdn.microsoft.com/en-us/library/system.windows.uielement(v=vs.110).aspx

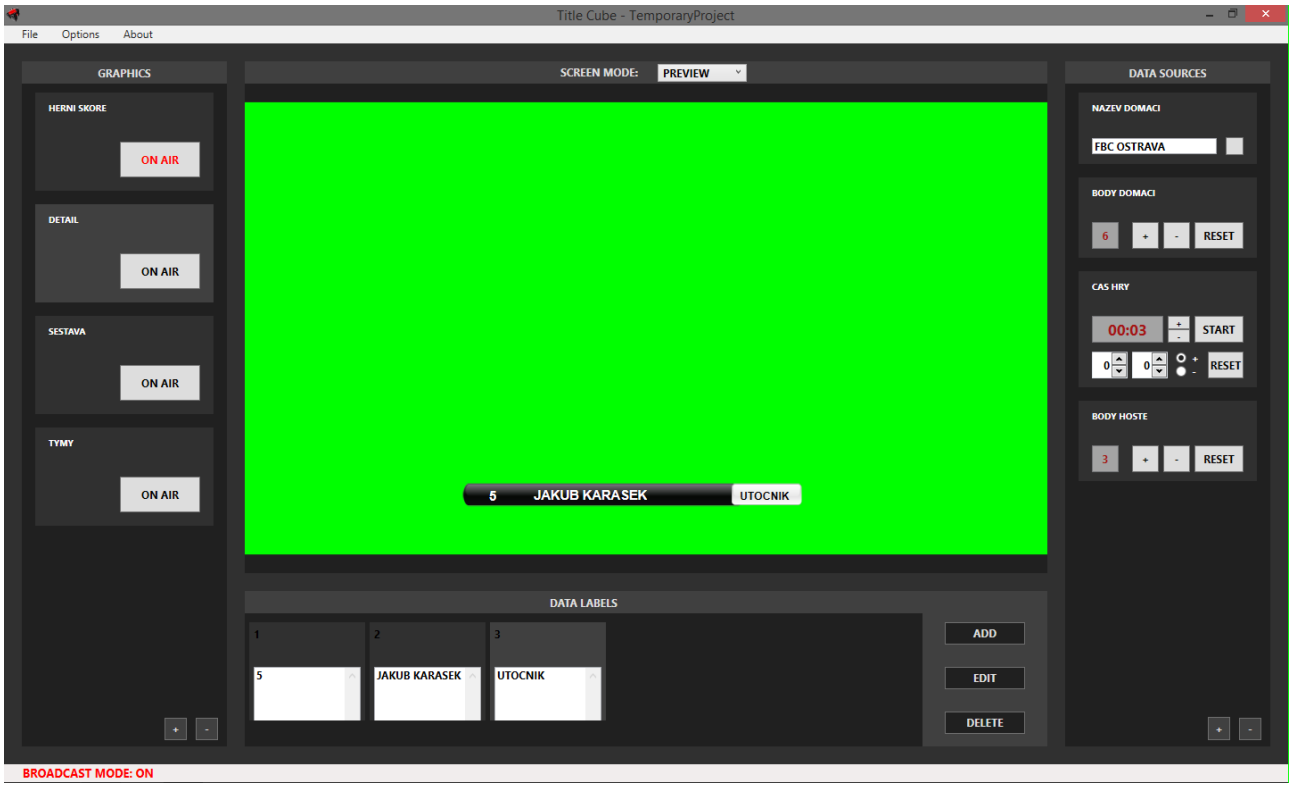
[11] http://msdn.microsoft.com/en-us/library/system.windows.controls.canvas(v=vs.110).aspx

[12] http://msdn.microsoft.com/en-us/library/system.windows.controls.viewbox(v=vs.110).aspx

13. Přílohy

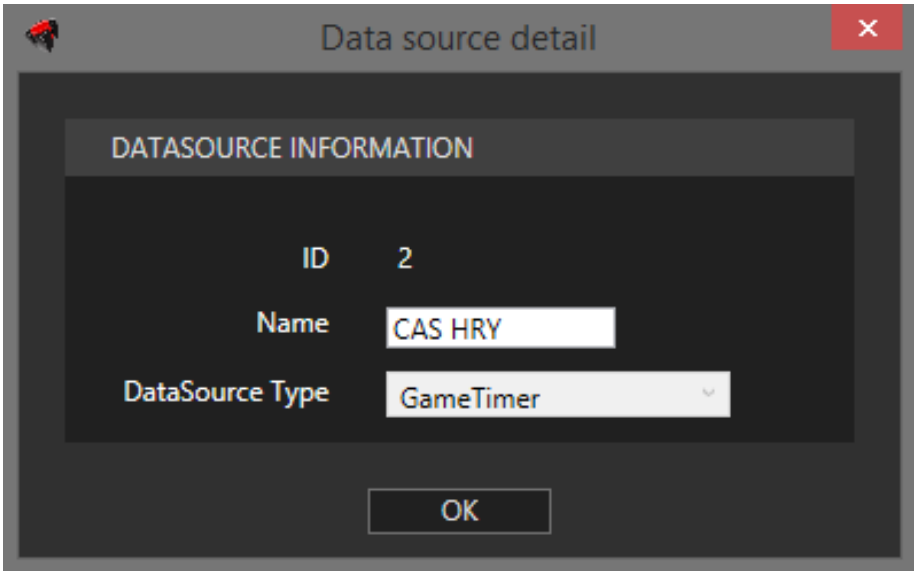
Příloha A: Ukázka výstupní obrazovky

Příloha B: Hlavní okno programu

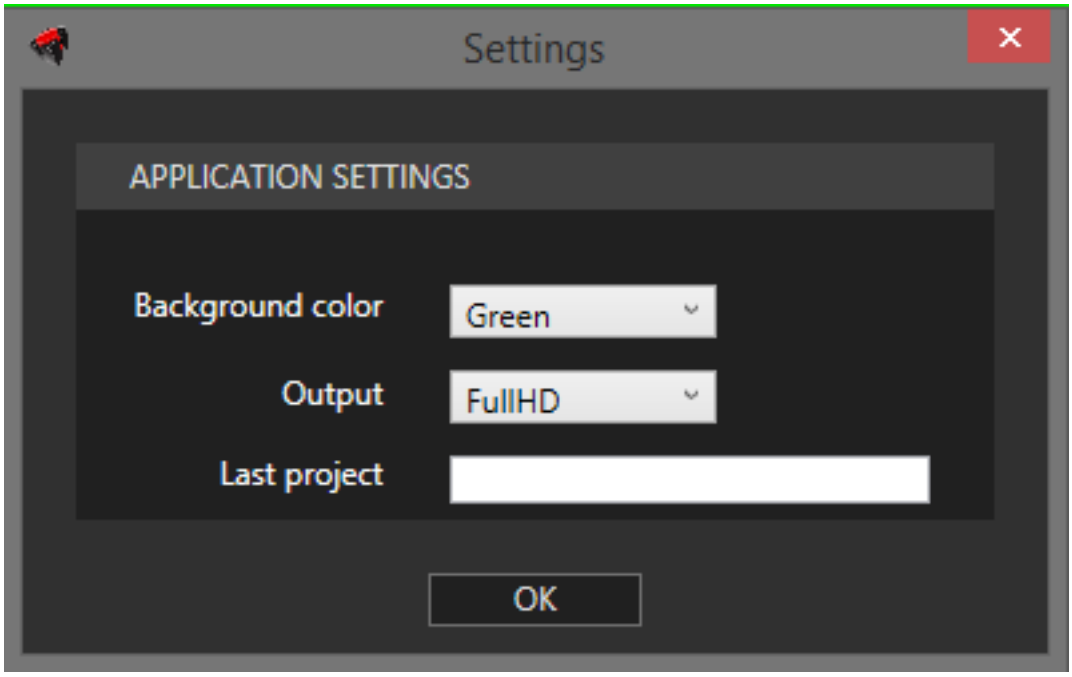


Příloha C: Okno DataLabelDetail

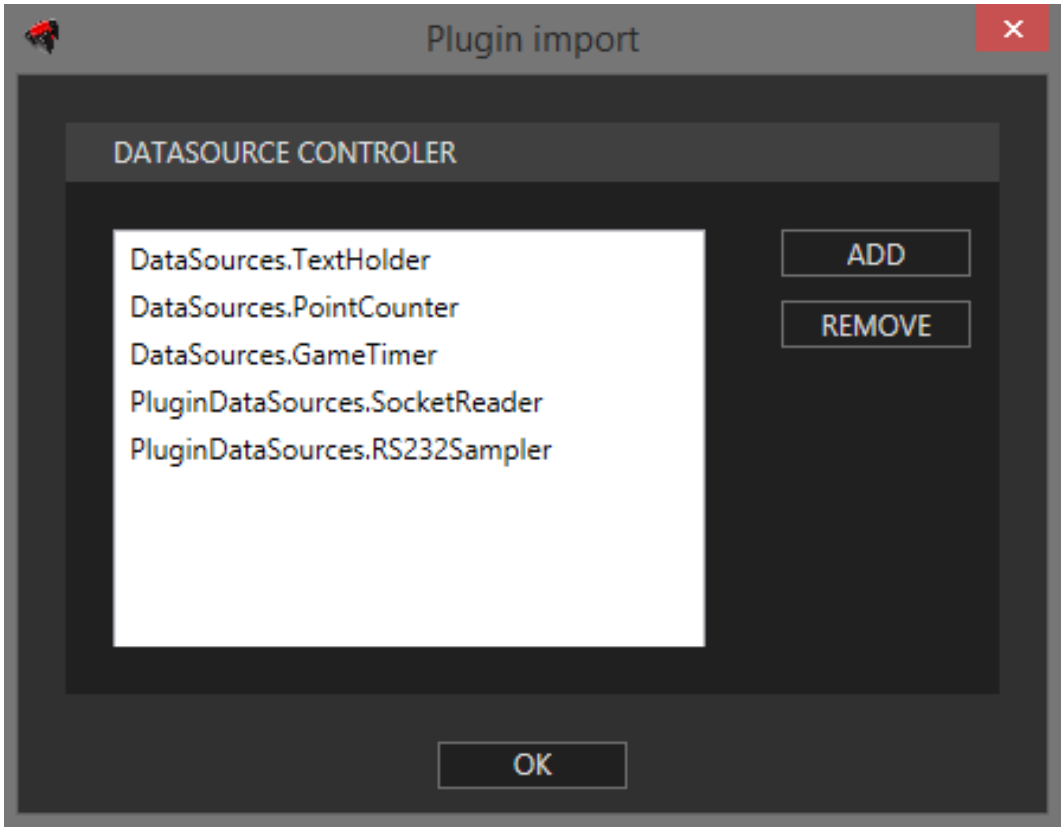
Příloha D: Okno DataSourceDetail



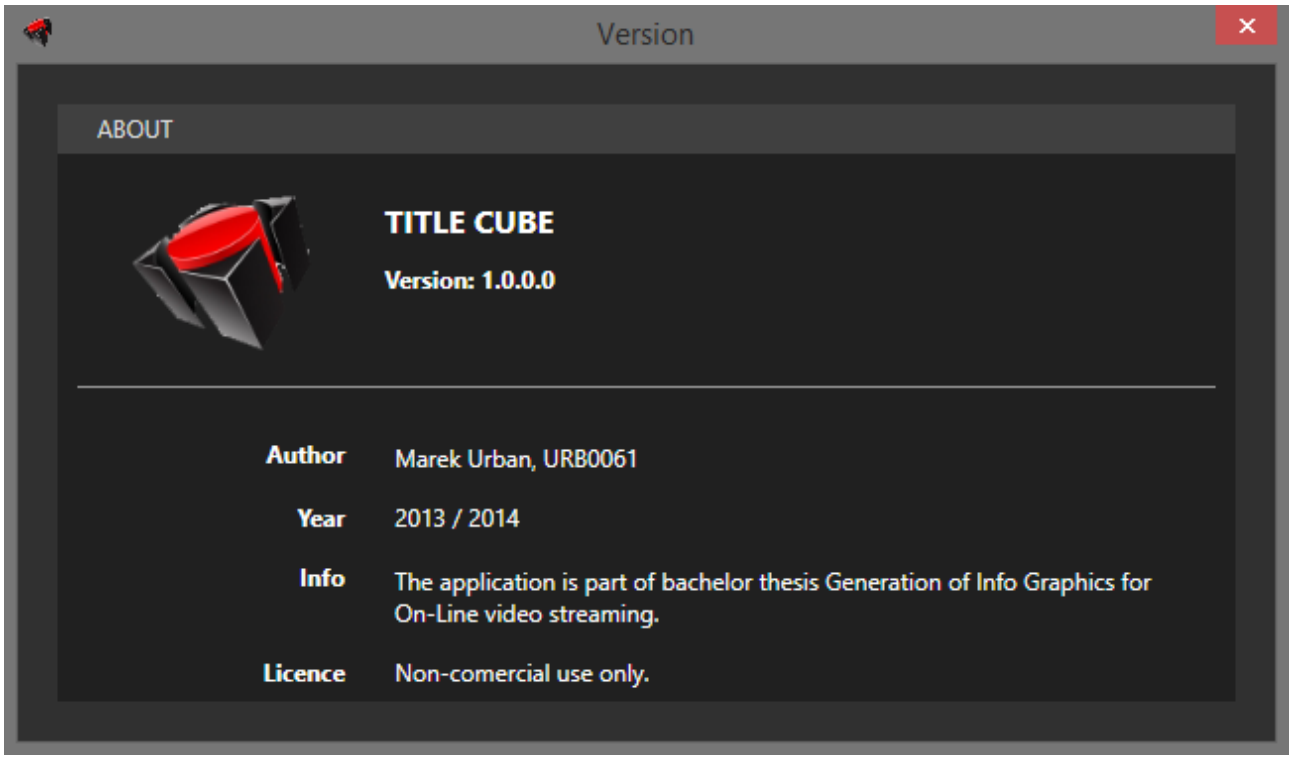
Příloha E: Okno ApplicationSettings



Příloha F: Okno PluginImport



Příloha G: Okno Version



Příloha H: Diagram Tříd